intel®

AB-39

APPLICATION BRIEF

Interfacing the Densitron LCD to the 8051

RICK SCHUE
REGIONAL APPLICATIONS SPECIALIST
INDIANAPOLIS, INDIANA

February 1996

Order Number: 270529-003

Information in this document is provided in connection with Intel products. Intel assumes no liability whatsoever, including infringement of any patent or copyright, for sale and use of Intel products except as provided in Intel's Terms and Conditions of Sale for such products.

Intel retains the right to make changes to these specifications at any time, without notice. Microcomputer Products may have minor variations to this specification known as errata.

*Other brands and names are the property of their respective owners.

†Since publication of documents referenced in this document, registration of the Pentium, OverDrive and iCOMP trademarks has been issued to Intel Corporation.

Contact your local Intel sales office or your distributor to obtain the latest specifications before placing your product order.

Copies of documents which have an ordering number and are referenced in this document, or other Intel literature, may be obtained from:

Intel Corporation
P.O. Box 7641
Mt. Prospect, IL 60056-7641

or call 1-800-879-4683

COPYRIGHT © INTEL CORPORATION, 1996

INTERFACING THE DENSITRON LCD TO THE 8051

CONTENTS

PAGE

INTRODUCTION 1

HARDWARE DESIGN 1

TIMING REQUIREMENTS 1

SOFTWARE 1

REFERENCES 1

INTRODUCTION

This application note details the interface between an 80C31 and a Densitron two row by 24 character LM23A2C24CBW display. This combination provides a very flexible display format (2x24) and a cost effective, low power consumption microcontroller suitable for many industrial control and monitoring functions.

Although this applications brief concentrates on the 80C31, the same software and hardware techniques are equally valid on other members of the 8051 family, including the 8031, 8751, and the 8044.

HARDWARE DESIGN

The LCD is mapped into external data memory, and looks to the 80C31 just like ordinary RAM. The register select (RS) and the read/write (R/W) pins are connected to the low order address lines A0 and A1. Connecting the R/W pin to an address line is a little unorthodox, but since the R/W line has the same set-up time requirements as the RS line, treating the R/W pin as an address kept this pin from causing any timing problems.

The enable (E) pin of the LCD is used to select the device, and is driven by the logical OR of the 80C31's RD and WR signals AND'ed with the MSB of the address bus. This maps the LCD into the upper half of the 64 KB external data space. If this seems a little wasteful, feel free to use a more elaborate address decoding scheme.

With the address decoding shown in the example, the LCD is mapped as follows:

Address	Function	Read/Write?
8000H	Write Command to LCD	Write Only
8001H	Write Data to LCD	Write Only
8002H	Read Status from LCD	Read Only
8003H	Read Data from LCD	Read Only
8004H to FFFFH	No Access	

Undefined results may occur if the software attempts to read address 8000H or 8001H, or write to address 8002H or 8003H.

TIMING REQUIREMENTS

The timing requirements of the Densitron LCD are a little slow for a full speed 80C31. The critical timing parameters are the enable pulse width (PW E) of 450 ns, and the data delay time during read cycles (tDDR) of 320 ns. The 80C31 is available at clock speeds up to 16 MHz, but at this speed these parameters are violated. Since the 80C31 lacks a READY pin, the only way to satisfy the LCD timing requirements is to slow the clock down to 10 MHz or lower. A convenient crystal frequency is 7.3728 MHz since it allows all standard baud rates to be generated with the internal timers.

SOFTWARE

The code consists of a main module and a set of utility procedures that talk directly to the LCD. This way the application code does not have to be concerned with where the LCD is mapped, or the exact bit patterns needed to control it. The mainline consists of a call to initialize the LCD, and then it writes a message to the screen, waits, and then erases it. It repeats this indefinitely.

The utility procedures include functions to initialize the display, send data and address to the LCD, home the cursor, clear the display, set the cursor to a given row and column, turn the cursor on and off, and print a string of characters to the display. Not all the functions are used in the software example.

REFERENCES

INTEL Embedded Controller Handbook, 210918

INTEL PL/M-51 User's Guide, 121966

DENSITRON Catalog LCDMD-C

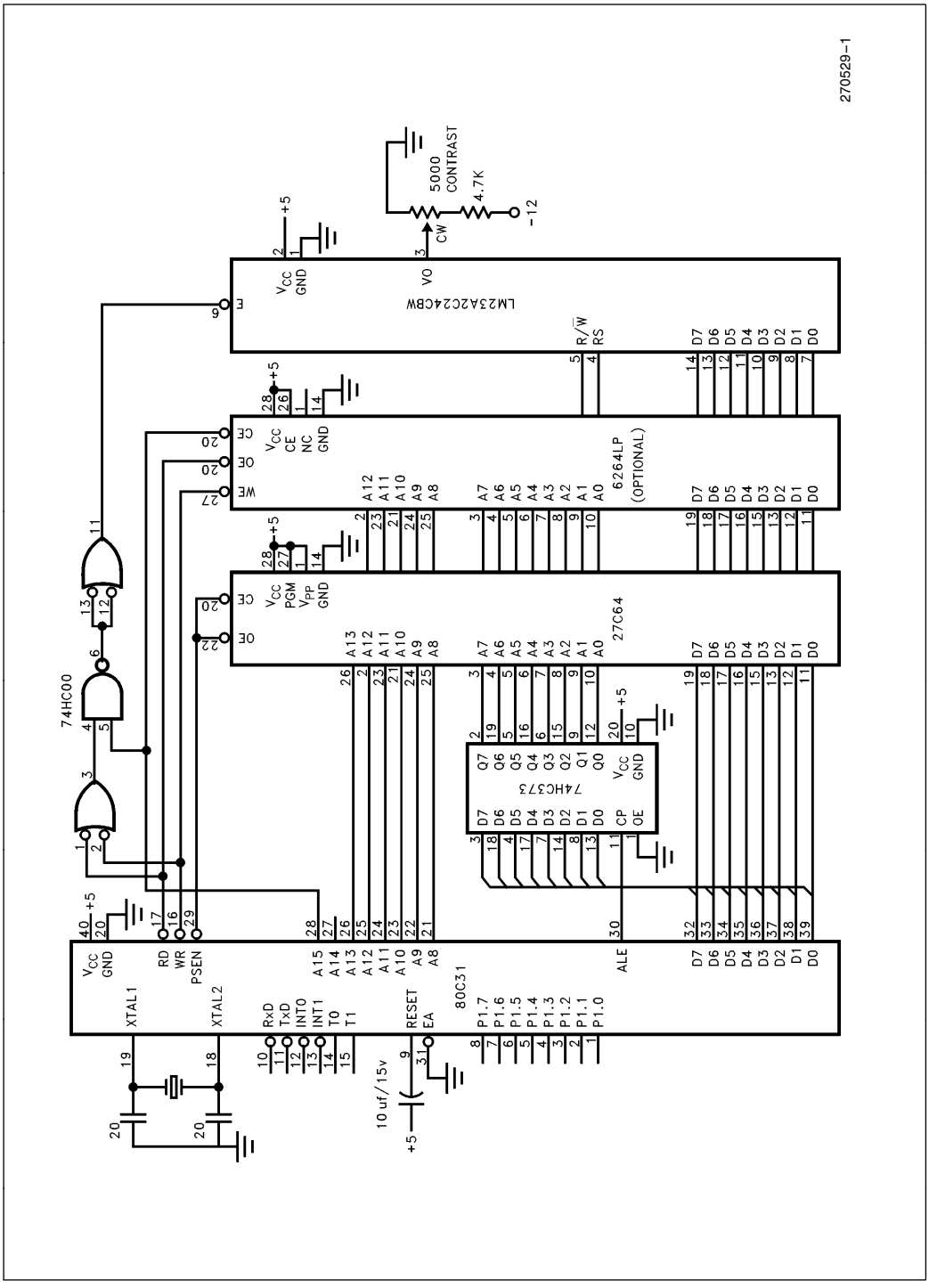

270529–1
5000
CONTRAST
4.7K
CW
+5
-12
LM23A2C24CBW
VCC
GND
VO
E
R/W
RS
D7
D6
D5
D4
D3
D2
D1
D0
6264LP
(OPTIONAL)
VCC
CE
NC
GND
OE
WE
A12
A11
A10
A9
A8
A7
A6
A5
A4
A3
A2
A1
A0
27C64
VPP
PGM
74HC00
74HC373
Q7
Q6
Q5
Q4
Q3
Q2
Q1
Q0
CP
OE
80C31
XTAL1
XTAL2
RD
WR
PSEN
ALE
RESET
EA
RxD
TxD
INT0
INT1
T0
T1
A15
A14
A13
P1.7
P1.6
P1.5
P1.4
P1.3
P1.2
P1.1
P1.0
10 uf/15v

Figure 1.

```
Main_module:  DO;

Delay: PROCEDURE (count) EXTERNAL;
  DECLARE    count            WORD;
END Delay;

Initialize_LCD: PROCEDURE EXTERNAL;
END Initialize_LCD;

Clear display: PROCEDURE EXTERNAL;
END Clear_display;

LCD_print: PROCEDURE EXTERNAL;
END LCD_print;

DECLARE LCD_buffer                (48)   BYTE   PUBLIC,
        sign_on_message            (*)   BYTE   CONSTANT
                ('INTEL 8051 DRIVES LCD - '
                 '2 ROWS BY 24 CHARACTERS '),
        i                                BYTE;

/* This is the start of the program */

  /* Initialize the LCD */
  CALL Initialize_LCD;
  CALL Clear_display;

  /* Now enter an endless loop to display the message */
  DO WHILE 1;

    /* Copy the message to the buffer */
    DO i = 0 to 47;
      LCD_buffer(i) = sign_on_message(i);
    END;

    /* Now print out the buffer to the LCD */
    CALL LCD_print;

    /* wait a while */
    CALL Delay(2000);

    /* now clear the screen */
    CALL Clear_display;

END;         /* of DO WHILE */

END Main_module;
```

Main Module

```
LCD_IO_MODULE: DO;

DECLARE LCD_buffer  (48)    BYTE      EXTERNAL,
        LCD_command         BYTE      AT (08000H) AUXILIARY,
        LCD_data            BYTE      AT (08001H) AUXILIARY,
        LCD_status          BYTE      AT (08002H) AUXILIARY,
        LCD_busy            LITERALLY '1000$0000B',
        i                   BYTE;

Delay: PROCEDURE (msec) PUBLIC;

  /* This procedure causes a delay of n msec */

  DECLARE msec                WORD,
          i                    WORD;

  IF msec > 0 THEN DO;
    DO i = 0 to msec - 1;
      CALL Time(5);         /* .2 msec delay */
    END;

END Delay;


LCD_out: PROCEDURE (char) PUBLIC;

  DECLARE char BYTE;

  /* wait for LCD to indicate NOT busy */
  DO WHILE (LCD_status AND LCD_busy) <> 0;
  END;

  /* now send the data to the LCD */
  LCD_data = char;

END LCD out;
```

LCD Driver Module

```
LCD_command_out: PROCEDURE (char) PUBLIC;

  DECLARE char BYTE;

  /* wait for LCD to indicate NOT busy */
  DO WHILE (LCD_status AND LCD_busy) <> 0;
  END;

  /* now send the command to the LCD */
  LCD_command = char;

END LCD_command_out;


Home_cursor: PROCEDURE PUBLIC;

  CALL LCD_command_out(0000$0010B);

END Home_cursor;


Clear_display: PROCEDURE PUBLIC;

  CALL LCD_command_out (0000$0001B);

END Clear_display;


Set_cursor: PROCEDURE (position) PUBLIC;

  DECLARE   position            BYTE;

  IF position > 47 THEN position = 47;
  IF position < 24 THEN CALL LCD_command_out(080H + position);
  ELSE CALL LCD_command_out(0C0H + (position - 24));

END Set_cursor;


Cursor_on: PROCEDURE PUBLIC;

  CALL LCD_command_out(0000$1111B);

END Cursor_on;


Cursor_off: PROCEDURE PUBLIC;

  CALL LCD_command_out(0000$1100B);

END Cursor_off;
```

LCD Driver Module (Continued)

```
LCD_print: PROCEDURE PUBLIC;

  /* This procedure copies the contents of the LCD_buffer
     to the display */

  CALL Set_cursor(0) ;
  DO i = 0 to 23;
    CALL LCD_out(LCD_buffer(i));
  END;
  CALL Set_cursor(24);
  DO i = 24 to 47;
    CALL LCD_out(LCD_buffer(i));
  END;

END LCD_print;


Initialize_LCD: PROCEDURE PUBLIC;

  CALL Delay(100);
  CALL LCD_command_out(38H);   /* Function Set */
  CALL LCD_command_out(38H);
  CALL LCD_command_out(06H);   /* entry mode set */
  CALL Clear_display;
  CALL Home_cursor;
  CALL Cursor_off;
  CALL Set_cursor(0);

END Initialize_LCD;


END LCD_IO_Module;
```

LCD Driver Module (Continued)